Internal Communication Protocol for Data Switching Equipment

[01] This application claims the benefit of U.S. Provisional Patent Application No. 60/236,165 filed September 29, 2000.

Field of the invention

[02] The invention relates to data switching, and in particular to methods of exchanging information internal to data switching equipment in operating thereof and provisioning data services.

Background of the invention

[03] In the field of data switching, the performance of data switching nodes participating in data transport networks is of outmost importance. The range of features supported and the array of deployable services is of an equally high importance.

[04] FIG. 1 is a schematic diagram showing general components of a data switching node 100. The data switching node 100 is a multi-ported device having a shared memory 102 design and forwarding Protocol Data Units (PDUs) between N physical ports 104 in accordance with supported data transfer protocols. Although pictured as such, the invention is not limited to data switching equipment having a shared memory design.

[05] PDUs include and are not limited to: cells, frames, packets, etc. Each PDU has a size. Cells have a fixed size while frames and packets may vary in size. Each PDU has associated header information used in forwarding the PDU towards a destination data network node in the associated data transport network. The header information is consulted at data switching nodes in determining the output port via which to forward the PDU towards the destination data network node. More than one output port may be determined to forward a PDU to as is the case for PDUs multicasted to a group of data network nodes.

[06] Each physical port 104 is adapted to receive and transmit data via an associated physical link 106 as shown at 108. Each physical port 104 has an associated physical data transfer rate. Physical port 104 designs having an adjustable physical data transfer rate exist. Physical ports 104 can also be used to convey data adhering to more than one data transfer protocol. A design requirement is that the data switching node 100 be able to process and convey PDUs such that all physical ports 104 receive and transmit simultaneously at their full physical data transfer rates. The operation of each physical port 104 is controlled via associated physical port operational parameters.

[07] The overall operation of a data switching node 100 includes: receiving at least one PDU via an input port 104, determining an appropriate output port 104 to forward the PDU to, scheduling the PDU for transmission, and transmitting the PDU via the determined output port 104. In determining the appropriate output port, PDUs may be stored in processing queues in the shared memory 102. Each physical port 104 has access to the shared memory 102 via a data bus 110. Processing queues are used to: match data transfer rates, match data transfer rates with processing rates, enable data flow rate estimation and enforcement, evaluate processing rates, enable statistics gathering, etc.

[08] Although in principle, the operation of data switching node 100 is simple, the implementation of such data switching equipment is not trivial. Efficient operation of the data switching node 100 is dependent on an efficient management of resources and efficient deployment of services.

[09] The header information is also consulted in: managing the operation of the data switching node 100, managing resources of the data switching node 100 and to some extent data network resources associated therewith, provisioning data services, etc.

[10] Managing the operation of the data switching node 100 and managing resources at the data switching node 100 is essential for efficient data switching performed by a switching processor 120. In forwarding PDUs, the switching processor 120 makes use of a PDU classifier 122 to inspect PDUs pending processing and to inspects a body of routing information provided by an associated destination address resolution function 124 to determine output ports 104. The PDU classifier 122 also makes a determination whether PDUs are unicast or a multicast.

[11] In processing PDUs, the body of routing information may be modified. Modifying the body of routing information is necessary in establishing new data transport routes in the data transport network for data sessions and the associated data transfers. As the header information is processed, the data switching node 100 learns of new data network nodes. In determining output ports 104 to forward PDUs to, the data switching node 100 learns of new routes to destination data network nodes.

[12] The body of routing information can be stored in the shared memory 102. The sharing of data storage resources for PDU buffering and for storing routing information, consolidates the memory storage requirements at the data switching node 100 and simplifies the design of the data switching node 100 leading to reduced implementation costs.

[13] An exemplary resource management function relates to data flow rate enforcement 130. Another resource management function and to some extent an operational management function of the data switching node 100 is provided via data flow statistics gathering 132. Link layer operation is managed through a port monitoring function 140.

[14] An example of a service includes virtual networking supported at the data switching node 100 via a virtual networking function 150. By deploying new services, the data switching node learns of new Virtual Local Area Network (VLANs) being established, etc. PDU header information is consulted to determine PDU treatment characteristics such as VLAN associativity in support of the virtual networking function 150, Class-of-Service (CoS) associativity in providing Quality-of-Service (QoS) guarantees in support of a Service Level Agreement (SLA) enforcement function 160, PDU forwarding priorities in support of data streaming functionality, etc.

[15] The learning ability of the data switching node is largely controlled by learning protocols which collectively provide data transport network topology discovery features. The learning ability is also limited by processing resources available at the data switching node such as processing power, memory storage resources, processing bandwidth, etc.

[16] Implementations of data switching equipment is under a high market pressure to reduce component count which led to embedded designs. Learning protocols used by the data switching nodes are typically embedded and form an integral part of the firmware/software executed during the operation of the data switching node.

[17] FIG. 2 is a schematic diagram showing an exemplary implementation of a data switching node providing resource management and delivering data services.

[18] Typical embedded designs of data switching equipment make use of a microcontroller 200 to enable an unmanaged operation thereof. The microcontroller 200 is factory programmed to support a specific feature set enabling the data switching node to function autonomously.

[19] The microcontroller 200 interfaces with the shared memory 102 to access PDU data and other data used in processing PDUs such as but not limited to the above mentioned body of routing information.

[20] In implementing the destination address resolution function 124, the microcontroller 200 interfaces with a Media Access Control (MAC) control database 210 storing tables specifying destination data nodes identifiers reachable via each physical port 104. The port monitoring function 140 may also use the same the MAC control database 210 to store operational parameters of each physical port 104 including data transfer rates, data transfer protocols supported, parameters associated with data network topology discovery, link status, etc. In support of the port monitoring function 140 the microcontroller 200 interfaces with each physical port 104 to obtain information, monitor and change operational parameters thereof.

[21] In implementing the data flow rate control function 130, the microcontroller interfaces with a PDU processing queue control block 220 and a rate control block 222. The PDU processing queue control block 220 tracks the usage of memory buffer space in the shared memory 102 via registers holding queue size values, queue locations in the shared memory 102, etc. The rate control block 222 stores registers specifying: queue low watermark occupancy levels, queue high watermark occupancy levels, current queue occupancy status for each processing queue, current data throughput rate for each processing queue, current data throughput status for each processing queue, data flow variables, whether flow control is enabled, etc.

[22] In implementing the virtual networking function 150, the microcontroller 200 interfaces with a VLAN index table 230 and a VLAN spanning tree database 232. The VLAN index table 230 stores associations between data network node identifiers and virtual network identifiers, data transmission priorities for virtual network associated data, data transmission parameters including bandwidths, etc. The VLAN spanning tree database 232 stores a hierarchical association between virtual networks and related parameters.

[23] In implementing the statistics gathering function 132, the microcontroller interfaces with a statistics counter block of registers 240 holding values for statistics counters. Examples of statistics counters include but are not limited to number of PDUs received, number of PDUs transmitted, number of bit errors instances, etc. There are global statistics counters, port statistics counters, service specific statistics counters, etc.

[24] The presented design can be reduced to a single silicon chip. Such embedded designs although providing for a very compact data switching node tend to be proprietary, not easily upgradeable, non-scaleable, etc. The supported feature set tends to be limited by what was typically required of data switching equipment at the time of development.

[25] For managed mode operation a management processor 250 may be used mainly for status reporting and higher level functions. The operational status of the data switching node is brought to the application layer and out to monitoring software applications such as network management applications perhaps, remote from the data switching node 100. A proprietary data connection 260 is provided between the microprocessor 200 and the management processor 250.

[26] In operation the microcontroller 200 and management processor 250 are used together to manage resources and deliver services. Functions of the embedded microcontroller 200 include: initialize internal and external memory registers, the coordination of information synchronization between the above mentioned components including the management processor 250, send commands to components for execution, receive reports from components, alert the management processor 250 component of operation critical events detected by other components, etc.

[27] Current designs have been implemented using a large number of memory access registers to reference different memory locations in enabling information exchange. In enabling the reporting of critical events, as many as 20 interrupt sources per physical port 104 have been used but, by provisioning access thereto via the above mentioned registers, the management processor 250 needs to reference multiple hierarchical registers to determine the interrupt source and service it.

[28] The microcontroller 200 is programmed with specific information exchange protocols for each component to enable the implementation of each function. The development of additional functionality and the addition of new features in support of enhanced and new services, necessitates the re-coding of the microcontroller 200. Any upgrade of any of the components including the management processor 250 also requires the re-coding of the microprocessor 200.

[29] Therefore there is a need to provide enhanced methods of monitoring the performance of data switching nodes as well as providing support for service delivery in enhancing the performance of the data switching node 100.

<u>Summary of the invention</u>

[30] In accordance with an aspect of the invention, a method of exchanging information internal to a data switching node is provided. The method includes a sequence of steps. The information to be exchanged is encapsulated in a data frame at a first component internal to the data switching node. The data frame is conveyed between the first component and a second component via a data exchange medium. The data is decapsulated by the second component.

[31] In accordance with another aspect of the invention, the exchanged information includes a data stream.

[32] In accordance with a further aspect of the invention, the exchanged information includes a request.

[33] In accordance with yet another aspect of the invention, the exchanged information includes an interrupt request.

[34] The conveyance of encapsulated information between internal components of data switching equipment enables easy expansion, upgrade and new deployment of features and services in data switching environments. In particular, it enables independent development of components, functions, features and services. The advantages are derived from a generic hardware implementation with a deployable, upgradeable and expandable feature set providing and enhancing support for current and future services.

Brief description of the drawings

[35] The features and advantages of the invention will become more apparent from the following detailed description of the preferred embodiment(s) with reference to the attached diagrams wherein:

FIG. 1 is a schematic diagram showing a general architecture of a data switching node;

FIG. 2 is a schematic diagram showing an exemplary prior art implementation of a data switching node providing resource management and delivering data services;

FIG. 3 is a schematic diagram showing an exemplary implementation of a data switching node providing resource management and delivering data services in accordance with an exemplary implementation of the invention;

FIG. 4 is a schematic diagram showing exemplary internal components of a data switching node and, associated requests and responses implementing an information exchange protocol in accordance with an exemplary implementation of the invention;

FIG. 5 is another schematic diagram showing exemplary internal components of a data switching node and, associated requests and responses implementing an information exchange protocol in accordance with an exemplary implementation of the invention;

FIG. 6 is yet another schematic diagram showing exemplary internal components of a data switching node and, associated interrupt requests implementing an information exchange protocol in accordance with an exemplary implementation of the invention; ; and

FIG. 7 FIG. 8, FIG. 9, FIG. 10 and FIG. 11 are a schematic diagrams showing data frame formats for data frames exchanged with the management processor in implementing an information exchange protocol in accordance with an exemplary implementation of the invention.

[36] It will be noted that in the attached diagrams like features bear similar labels.

Detailed description of the embodiments

[37] In accordance with a preferred embodiment of the invention the development of the application code and the development of the hardware components of a data switching node are separated. The separation is enabled via a unified information exchange protocol shielding a management processor from hardware implementation details.

[38] The unified information exchange protocol includes a group of data frame encapsulated interrupt requests, requests and responses to enable management of data switching node resources and service delivery.

[39] FIG. 3 is a schematic diagram showing an exemplary implementation of a data switching node providing resource management and delivering data services in accordance with an exemplary implementation of the invention.

[40] The unified information exchange protocol makes use of data frames exchanged between the management processor 300 and a frame translator 310 via a data exchange medium 320. The data exchange medium 320 may include, but is not limited to: a data bus having data bus width of: 8 bits, 16 bits, 32 bits, etc.; as well as a serial link. For the purpose of exchanging information via the data exchange medium 320, the data frames exchanged are divided in to data fragments having a length corresponding to the width of the data exchange medium 320.

[41] The frame translator 310 is associated with an interface 330 which communicates directly with the different components implementing the features and associated functions of the data switching node. The interface 330, the translator 310 and the unified information exchange protocol, mask low level information exchange implementation details of service/feature delivery components from service/feature enabling components such as the management processor 300.

[42] Although only one service enabling component such as the management processor 300 is shown, the invention, through the unified information exchange protocol, can be extended to multiple service enabling components, each of which is optimized in provisioning specific services.

[43] FIG. 4, is a schematic diagram showing exemplary internal components of a data switching node and, associated requests and responses implementing an information exchange protocol in accordance with an exemplary implementation of the invention.

[44] The management processor 400 is adapted to exchange information such as a stream of data with service delivery components of the data switching node. Data stream information exchanges may be used in initializing service delivery components on startup and/or in synchronizing information stored therein with the management processor 400.

[45] In accordance with a preferred implementation of the invention, conveyed data streams are divided into data granules of up to 32 bytes. The invention is not limited to 32 byte long data granules, the granule size being a design choice. Each data granule is encapsulated as a payload of a data frame. An 8 byte header is included in the data frame prior to the conveyance thereof.

[46] The header specifies the associativity of the conveyed data granule with one of a group of data frame types. The group of data frame types includes a request for a memory write 410 from the management processor 400, a request for a memory read 420 from the management processor 400, and a read complete response 430 sent to the management processor 400. The payload of the read complete response 430 may also correspond to a data granule of a data stream to be conveyed to the management processor 400.

[47] The data frame formats corresponding to requests 410, 420 and response 430 are presented in FIG. 7.

[48] FIG. 5 is another schematic diagram showing exemplary internal components of a data switching node and, associated requests and responses implementing an information exchange protocol in accordance with an exemplary implementation of the invention.

[49] In support of the exemplary address resolution function 124 and the virtual networking function 150, the management processor 500, in maintaining the MAC control database 210 service delivery component and in optimizing the performance of the PDU classifier 122 service delivery component, issues data frame encapsulated requests 510 such as: a learn MAC address request, a delete MAC address request, a search MAC address request, a learn multicast

address request, delete multicast address request, search multicast address request, etc.

[50] In response to the search MAC address request and the search multicast address request, the management processor 500 receives data frames encapsulating responses 520: a response to a search MAC address request and a response to a search multicast address request.

[51] The above mentioned service delivery components in performing their respective functionality issue requests 530 for the management processor including: a learn MAC address request, a delete MAC address request, a delete multicast address request, a new VLAN port request (announcement), an age VLAN port request (announcement), etc. (Aging features are typically used in minimizing memory storage requirements at the data switching node by deleting stale information not use for a relatively long period of time.)

[52] The data frame formats corresponding to requests 510, 530 and response 520 are presented in FIG. 8, FIG. 9 and FIG. 10.

[53] FIG. 6 is yet another schematic diagram showing exemplary internal components of a data switching node and, associated interrupt requests implementing an information exchange protocol in accordance with an exemplary implementation of the invention.

[54] In support of the port monitoring function 140 and the statistics gathering function 132, the management processor 600 is informed of detected critical events via encapsulated interrupt requests 610. An interrupt request on physical link state change is issued by physical ports 104 to inform the management processor 600 whether the link is functional. An interrupt request on statistic counter roll-over is sent to the management processor 600 each if the value of a cumulative statistic counter exceeds a maximum expressible value of a register holding the value of the cumulative statistic counter. (Concurrent with the issuance of the statistic counter roll-over interrupt request the value of

the register associated with the statistic counter is reset to a predetermined value – typically 0.)

[55] The data frame format corresponding to interrupt requests 610 is presented in FIG. 11.

[56] As mentioned above an 8 byte header is used in conveying data frames. The header includes data fields specifying: a data frame type identifier, a data frame sequence number, a memory address for read and writes, etc. The invention is not limited to the above data fields other fields may be used in implementing different features and functionality. At least the data frame type identifier data field is mandatory. Data fields in the header have a specific location with respect to the start of the data frame. In the example shown the data frame type identifier is specified in the fist data field and specifically the first 4 bits of the first byte (top right corner). At a minimum, the data frame type identifiers have to be unique for data frames sent via the data exchange medium 320 in a single direction; that is, data frame type identifiers may be reused for data frame transfers in the opposite direction.

[57] Data fields are also used in exchanging information. Various data fields are shown in the above presented diagrams each of which has a specific location with respect to the beginning of the data frame. In conveying information in accordance with the invention, less than the full payload of each data frame may be used. The data exchange medium 320 may make use of hardware handshaking data flow control specifying the beginning and the end transmission of relevant information held in each data frame. The use of hardware handshaking in conveying partial data frames optimizes the use of the bandwidth of the data exchange medium 320.

[58] The embodiments presented are exemplary only and persons skilled in the art would appreciate that variations to the above described embodiments may be made without departing from the spirit of the invention. The scope of the invention is solely defined by the appended claims.